VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

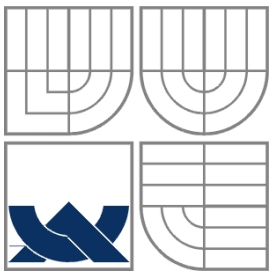
SLEDOVÁNÍ ODEZEV SÍŤOVÝCH SLUŽEB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

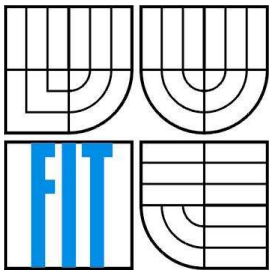
AUTOR PRÁCE
AUTHOR

PETER SAGÁL

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SLEDOVÁNÍ ODEZEV SÍŤOVÝCH SLUŽEB

NETWORK AND APPLICATION RESPONSE TIME MONITORING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETER SAGÁL
AUTHOR

VEDOUCÍ PRÁCE — ING. JIŘÍ TOBOLA
SUPERVISOR

BRNO 2010

Abstrakt

Táto bakalářská práce se zabývá analýzou požadavků, návrhem a popisem implementace aplikace na monitorování odezev sítě, síťových prvků a aplikací. Práce představí dvě základní metodiky sledování a analýzy síťových odezev – pasivní a aktivní monitorng, na základě kterého pracuje navrhnutá aplikace, která poskytuje webové rozhraní pro správu a prezentaci výsledků monitorování.

Abstract

This bachelor‘s thesis deals with analysis of requirements, design and implementation of application for monitoring of network, network components and application‘s response time, network components and applications. Document will introduce two basic methods of monitoring and analysis of network responses - passive and active monitorng, which is the basis of suggested application that provides a web interface for management and presentation of monitoring results.

Klíčová slova

Monitorování sítě, aktivní monitoring, pasivní monitoring, démon, skript

Keywords

Network monitoring, active monitoring, passive monitoring, daemon, script

Citace

Sagál Peter: Sledování odezev síťových služeb, bakalářská práce, Brno, FIT VUT v Brně, 2010

Sledování odezev síťových služeb

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Toboly
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Peter Sagál
19.5.2010

Poděkování

Chcel by som poďakovať v porvom rade rodičom, za poskytnutie možnosti študovať vysokú školu podľa vlastného výberu, a vedúcemu bakalárskej práce Ing. Jiřímu Tobolovi za poskytnutie námetu.

Obsah

1 Úvod

Dostupnosť služieb – v dnešnej dobe najviac používaná veličina reprezentujúca kvalitu aplikácie je hlavným cieľom každého poskytovateľa a prevádzkovateľa informačných technológií. Každý aj nepatrný výpadok môže mať ďalekosiahle negatívne následky, ktoré sa nemusia prejaviť ihneď v naväznosti na nedostupnosť služby. Od krátkeho výpadku elektronického obchodu, ktorý spôsobí prepad tržieb až po globálny kolaps leteckej dopravy v prípade že by prestal fungovať navigačný a komunikačný systém. Dostupnosť a rýchlosť odozvy aplikácií je úzko spätá so službami ktoré aplikáciu poskytujú – napríklad webový server v spojení s databázovým serverom, emailové služby, súborové služby atd. Preto je nutné na chod týchto služieb dohliadať, aby bolo možné predchádzať výpadku prípadne čo najrýchlejšie odhlaliť jeho príčinu aby bolo možné výpadok v čo najkratšej dobe eliminovať.

Väčšinu výpadkov aplikácií stále odhaľujú až koncoví uživatelia v momente keď nemôžu pristupovať k požadovaným službám. To je bohužiaľ najhorší prípad vzhľadom na ich dôveru k poskytovaným službám – preto je nutné zavádzať sledovanie týchto aplikácií do každej sféry IT. Stale viac firiem a poskytovateľov služieb investuje do technológií, ktoré sa vysporiadávajú s dôsledkami výpadku (napríklad pravidelné zálohovanie dôležitých dát), ale sledovaniu aktuálneho stavu svojich aplikácií nevenujú veľkú pozornosť. Nechcem týmto zmenšovať dôležitosť zálohovania, ale monitoring by mal byť na rovnakej úrovni dôležitosti ako zálohovanie.

Nespornou výhodou monitorovanej aplikácie je možnosť v reálnom čase sledovať rýchlosti reakcií na uživateľské vstupy, porovnávanie v rámci dlhodobého priemeru a v prípade väčšej odchylky ihneď podniknúť akcie na presnú detekciu a odstránenie chybového stavu (chyba nemusí byť len nedostupnosť ale aj dlhý reakčný čas). Základom pre odstránenie problému je lokalizácia jeho príčiny a práve túto príčinu pomáha monitoring odhaľovať.

Témou tejto bakalárskej práce je porovnať možnosti monitorovania sieťových služieb a aplikácií z pohľadu získavania dát – aktívne a pasívne monitorovanie siete. Práca je zameraná predovšetkým na aktívne monitorovanie a z toho vychádza aj praktická časť.

Cieľom práce je návrh a implementácia systému schopného monitorovať základné sieťové služby s možnosťou ďalšieho rozšírenia a nasadenia do produkčnej sféry.

Práca popisuje dve varianty sledovania siete, preberá ich vzájomné výhody, nevýhody a detailnejšie popisuje aktívne monitorovanie. Ďalej popisuje návrh systému vykonavajúceho aktívne monitorovanie, poisuje jeho implementáciu a diskutuje jeho aktuálne výhody a nedostatky, obsahuje návrhy na možné vylepšenia a rozšírenia.

2 Monitorovanie sieťových služieb a aplikácií

2.1 História

Základným prostriedkom pre monitorovanie stavu siete na báze TCP/IP a jej prvkov je jeden z IP protokolov – ICMP (Internet Control Message Protocol) popísaný v roku 1981 v dokumente RFC 972. Jeho základnou úlohou je poskytovanie informácií o chybových stavoch pri prenose datagramov. V prípade chyby pri doručovaní datagramu vyšle sieťový prvok odosielateľovi paketu ICMP správu inforumujúcu o tejto skutočnosti – napr. Destination unreachable alebo Time exceeded, v správe je väčšinou zahrnutá hlavička (prípadne celé telo) pôvodného paketu [6].

Ďaľším zo sady IP protokolov je SNMP (Simple Network Management Protocol) popísaný v RFC 1157 poskytujúci unifikovaný spôsob prenášania štatistických údajov zbieraných zo sieťových prvkov. V rámci SNMP protokolu medzi sebou komunikuje správca (manager) a agent. Správca zasiela požiadavok na dáta a agent zasiela správcovi odpovede s hodnotami požadovaných parametrov [5].

Tieto dva protokoly ale neboli navrhnuté na detailné monitorovanie služieb poskytovaných cez sieť. Preto boli postupom času vyvíjané sofistikovanejšie aplikácie schopné monitorovať široké škály aplikácií a ich parametrov. Samotné monitorovanie sa delí podľa spôsobu na dve skupiny: aktívne a pasívne.

2.2 Pasívne monitorovanie

Pasívne monitorovanie, ako už názov napovedá, je založené na odchytávaní a analýze sieťovej prevádzky (paketov) bez interakcie s monitorovaným subjektom. K tomuto účelu sa používajú vo väčšine prípadov NMIC (Network Monitoring Interface Card) karty – sú veľmi podobné obyčajným sieťovým kartám, avšak na rozdiel od nich nemajú žiadnu fyzickú MAC adresu, neodosielajú do monitorovanej siete žiadne dáta a ich prevádzka je čisto transparentná (tj. sú neviditeľné pre ostatné zariadenia pripojené k sieti). NMIC väčšinou bývaju priamo integrované do aktívnych sieťových prvkov (switche, routery). Chovanie NMIC sa dá napodobiť na väčšsine obyčajných sieťových kariet pomocou tzv promiskuitného režimu, kedy karta prijíma všetky pakety ktoré prichádzajú po sieti (tj nezahadzuje pakety, ktoré nie sú určené pre jej MAC adresu) a predáva ich špecilaizovanej aplikácii zvanej sniffer (napr. tcpdump, Wireshark).

2.2.1 Využitie pasívneho monitorovania

Monitorovanie sieťovej prevádzky pasívnym spôsobom sa vo väčšine prípadov používa na analýzu prenášaných dat za účelom účtovania množstva dát prenesených sieťovým segmentom. Vzhľadom na to, že sa zachytávajú reálne dáta generované sieťovými službami, je možné podľa nich zistit reakčné časy medzi požiadavkom a odpoveďou, priepustnosť a vyťaženie sieťových služieb, prípadne ich nedostupnosť. Bohužiaľ ale väčšina chybových stavov je odhalených až po niekoľkonásobnom zlyhaní zachyteného požiadavku (tj až po vzniku problému), kedy už má tento výpadok priamy vplyv na koncového uživateľa (klienta) vysielajúceho požiadavky, ktoré nie sú obslúžené.

Pasívne monitorovanie z tohoto dôvodu nie je používané na monitorovanie dostupnosti služieb, avšak je veľmi silným nástrojom na zisťovanie výkonnosti systému, pretože má k dispozícii reálne dáta ktoré systém obsluhuje – na základe množstva požiadavkov presne zisťuje reakčné časy prípadne straty požiadavkov a tým pádom je schopné predpovedať maximálnu možnú zaťažiteľnosť systému v daných podmienkach alebo poskytnúť podklady pre optimalizáciu systému pre väčšiu záťaž.

2.3 Aktívne monitorovanie

Akvíne (syntetické) monitorovanie je založené na napodobovaní (simulácii) uživateľských požiadavkov, ich odosielaní a analýze odpovedí aplikácie (reakčný čas, rýchlosť prenosu dát, obsah samotnej odpovede). Systém automatizovane posiela požiadavky aplikáciám a analyzuje odpovede, na základe výsledkov potom zostavuje štatistiku. V prípade odchylky od dlhodobého priemeru prípadne výpadku upozorňuje správcu aplikácie na podozrivý stav (predĺženie odoziev, strácanie požiadavkov,...)

Aktívne sledovanie siete dokáže odhaliť nastávajúci problém ešte pred tým ako dôjde k neobslúženiu požiadavku od klienta – to je hlavným dôvodom prečo sa používa na monitorovanie dostupnosti kritických služieb. Monitorovací systém posiela monitorovanej aplikácii požiadavky simulujúce reálne správanie systému – hlavným problémom je správne oddeliť syntetické požiadavky od reálnych, aby nenastala nekonzistencia stavu sledovanej aplikácie v dôsledku konfliktu reálnej transakcie s monitorovacou. Najlepším príkladom je monitorovanie funkčnosti databázoveho servera, kedy monitoring v pravidelných intervaloch spúšťa na serveri príkaz vykonávajúci prednastavenú akciu. Tento príkaz musí byť postavený tak aby neazasahoval do produkčných dát a zároveň aby čo najlepšie kopíroval reálne požiadavky klientov.

Hlavnou nevýhodou je fakt, že syntetický test vo väčšine prípadov nemôže simulovať úplne každú situáciu, ktorá môže nastať v reálnom nasadení – nemusí včas odhaliť každú možnú špecifickú závadu a je dosť zložité vytvoriť testovacie požiadavky, ktoré sú posielané monitorovaným službám (napr. SQL dotazy)

2.3.1 Monitorovanie dostupnosti sieťového prvku, portu

Prvým a najzákladnejším spoôsobom sledovania používaným v produkčnej sfére je monitorovanie dostupnosti sieťováho prvku na IP vrstve. Využíva sa pri tom ICMP protokol a jeho pakety so správami `Echo request` a `Echo reply`. Týmto spôsobom (za predpokladu neblokovaného ICMP protokolu) získava monitoring informácie o dostupnosti alebo nedostupnosti daného prvku a dĺžke času, ktorý sieť potrebuje pre doručenie odpovede.

Druhým spôsobom používaným pre zisťovanie dostupnosti a reakčného času hostu je posielanie požiadavkov na pripojenie na zvolený port v rámci zvoleného protokolu (TCP/UDP). Monitoring meria čas za ktorý sa vytvorí spojenie na danom porte. Týmto spôsobom sa zároveň vykonáva základná kontrola sieťovej služby – či služba na danom porte naslúcha (prijíma spojenia). Nie je však možné overiť o akú službu sa jedná (napr. na porte 80 väčšinou naslúcha www server, ale môže na ňom byť spustená aj hocijaká iná služba prijímajúca tcp spojenia). Preto je na dôkladnejšie testovanie aplikácie použiť prepracovanejšiu metódu.

2.3.2 Aplikačné monitorovanie

Najzložitejším spôsobom monitorovania dostupnosti aplikácie je simulovanie reálnych užívateľských požiadavkov na danú službu. Týmto spôsobom už sa overuje prítomnosť a zároveň funkčnosť danej aplikácie na sledovanom systéme, rýchlosť jej reakcií a samotnú schopnosť správne a včas odpovedať na zvolené požiadavky. Príkladom takéhoto monitoringu môže byť sťahovanie súboru z webového serveru, pripájanie a prenos súborov na FTP server, pripájanie k terminálovým službám a spúšťanie príkazov, vykonávanie SQL požiadavkov. Tento spôsob monitorovania poskytuje narozdiel od predošlého kvalitnejšie výsledky, avšak je zložitejší na implementáciu a náročnejší na prevádzku.

Aplikačné monitorovanie je najvhodnejší spôsob zisťovania výkonnosti aplikácie, jej dostupnosti a schopnosti reagovať na požiadavky. Simuluje reálne požiadavky klientov, ktoré sa najčastejšie vyskytujú v systéme. Dokáže odhaliť najviac chybových stavov ktoré môžu nastať, či už v aplikácii alebo sieťovom spojení. Výpadok zistený ICMP paketom ešte nemusí znamenať nedostupnosť služby na serveri, pretože ICMP nepoužíva spojenie zabezpečené voči stratám paketov.

2.3.3 Plánovanie monitoringu

Veľmi dôležitou súčasťou sledovania sieťových služieb je správne plánovanie testov v časovom období. Väčšina služieb poskytovaná v sieti je typu klient-server, na ktoré sa pripájajú reálni zákazníci (klienti) počas produkčnej doby (napr. 9-20 hod.) cez deň. To znamená, že monitoring v tomto časovom úseku by mal byť dôkladnejší (napríklad častejšie plánované kontroly funkčnosti) ako v čase keď služba nie je zaťažovaná reálnymi požiadavkami – monitoring by mal kopírovať reálnu úroveň zaťaženia monitorovaného systému.

Druhou dôležitou podmienkou je to, aby monitoring nespôsoboval nadmernú záťaž svojou činnosťou vzhľadom na záťaz generovanú reálnymi požiadavakami, ktoré služba obsluhuje. Podľa tejto podmienky sa volí časový interval medzi samotnými monitorovacími požiadavkami, ktoré monitoring posiela službe. Ten musí byť zvolený tak, aby nevznikla možnosť, že kritický výpadok nebude odhalený a zároveň monitoring nebude zbytočne zaťažovať monitorovanú službu a tým spôsobené oneskorenie obsluhy reálnych klientov.

3 Návrh aplikácie

Táto kapitola obsahuje analýzu požiadavkov na systém monitorujúci časové odozvy siete, sieťových prvkov a služieb poskytovaných cez sieť na základe aktívneho monitoringu. Ďalej obsahuje návrh implementácie tohoto systému a špeficikácie použitých sieťových a aplikačných protokolov.

3.1 Analýza požiadavkov na aplikáciu

Od systému je požadované aby zvládal monitorovanie širokej škály aplikácií a poskytoval užívateľovi prehľadné webové rozhranie s prezentáciou výsledkov a možnosťami nastavenia monitorovania. Dalej by system mal byt rozsiritelny o nove funkcie, pripadne sposoby monitorovania.

Aby systém čo najviac splňoval podmienku, že jeho požiadavky na služby budú podobné tým reálnym, na samotne dotazovanie služieb je lepšie využívať uz navrhnuté a bežne používané programy pokiaľ to povaha testovania dovoľuje (napríklad webový klient, icmp ping, mysql klient). V prípade, že je nutné použit klientskú aplikáciu vlastnú, táto musí splňovať zadané špecifikácie protokolu nad ktorým bude pracovať.

Zobrazenie výstupov programu (tj štatistiky získané monitorovaním) je najlepšie grafickou formou, vďaka ktorej sa dá rýchlo odhaliť výchylka v reakčnom čase systému. Na základe tohoto požiadavku bude najlepšie použiť aplikáciu priamo k tomuto účelu vyvinutú a používanú.

Možnosť rozšíriť aplikáciu o ďaľšie funkcie a spôsoby monitorovania je dôležité uvážiť pred vývojom samotnej aplikácie. Od tohoto požiadavku sa bude odvíjať celá navrhnutá architektúra, ktorá bude obsahovať rozhranie cez ktoré je možné do aplikácie pridávať ďaľšiu funkčnosť.

3.1.1 Napodobenie reálnych požiadavkov

Aplikácia má za úlohu podporovať obidve spomenuté úrovne pre aktívne monitorovanie. Tj. monitorovanie dostupnosti sieťového prvku a portu. K tomuto účelu je najlepšie použiť univerzálny nástroj na ICMP testovanie siete poskytovaný operačným systémom – aplikáciu ping. Na monitorovanie dostupnosti portu existuje viac aplikácií, ktoré túto úlohu vykonávajú rôznymi spôsobmi, avšak z hľadiska implementácie je rozumné použiť nástroj vlastný, ktorý bude vytvorený len k tomuto účelu a bude používať stále jeden spôsob. V opačnom prípade by mohli vznikať odchýľky spôsobené inou metodikou pripájania na port.

Rovnaká je situácia aj v prípade aplikačného monitorovania, kde sa väčšina monitorovaných služieb dá monitorovať pomocou zaužívaného univerzálneho klienta (pokial je tento vhodný na vykonávanie monitoringu, napríklad aplikácia wget alebo ftp), ale na simuláciu niektorých požiadavkov je lepšie vytvoriť špecializovaných klientov, presne zodpavedajúcich požiadavkom pre

monitorovanie danej služby (POP3, SMTP). Ako príklad uvádzam monitorovanie dostupnosti webového serveru, kedy je najlepšie použiť rozšírenú aplikáciu wget (ktorú obsahuje väčšina unixových systémov ako systémovú aplikáciu, prípadne ju poskytuje ako príslušenstvo) ktorá na základe zadaných parametrov pošle požiadavok webovému serveru, spracuje jeho odpoveď a prijme súbor poskytnutý webovým serverom. Na druhú stranu v prípade POP3 protokolu je situácia iná. Na monitorovanie tejto služby je lepšie vytvoriť klienta vlastného, ktorý bude vždy používať rovnakú sadu a postupnosť príkazov protokolu, tak aby bolo možné porovnávať dĺžku odozvy serveru.

3.1.2 Analýza a správa nameraných hodnôt

Monitorovacia aplikácia nemusí len posielať požiadavky na odozvy služieb, ale tiež musí prijaté odpovede analyzovať a správne interpretovať namerané hodnoty. Aby to bolo možné, musí mať presne definovaný spôsob spracovania týchto hodnôt. Ako najlepšia voľba na vykonávanie tejto funkcie bol zvolený program RRDTool, ktorý je pre tento účel navrhnutý a vyvíjaný. Obsahuje možnosť ukladať namerané dáta do štatistickej databázy s presne špecifikovanou vnútornou štruktúrou ukladania cyklických dát. Táto databáza obsahuje dôležitú vlastnosť a tou je stálosť svojej veľkosti nezávisle od dat ktoré obsahuje. Na druhú stranu nevýhodou je obmedzené množstvo dát ktoré môže obsahovať – preto je veľmi dôležité určiť správnu štruktúru a rozsah dát ktoré bude databáza obsluhovať. Jedným z parametrov je dĺžka časového obdobia obsiahnutého v databáze, po jej naplnení sa s každou aktualizáciou odstráni najstarší záznam v databáze a tak sa celková veľkosť súboru s databázou udržuje na konštantnej veľkosti.

3.1.3 Zobrazenie výsledkov

Spomenutá aplikácia RRDTool obsahuje taktiež nástroj na tvorbu prehľadných štatistických grafov na základe dát obsiahnutých v databáze. Toto je ďalším dôvodom pre zvolenie tohoto programu ako poskytovateľa dátového úložiska pre štatistické údaje získané monitoringom. Nie je tým pádom nutné používať jeden program na správu dát a druhý na generovanie výstupov pre uživateľa.

3.1.4 Webové uživateľské rozhranie

Aplikácia má poskytovať webové rozhranie pre interakciu s uživateľom, pomocou ktorého bude možné vykonávať nastavenia monitoringu a zobrazovať výsledky meraní. Na tento účel sa používajú webové skriptovacie jazyky, ktoré generujú dynamické webové stránky na základe požiadavkov uživateľa. Ako skriptovací jazyk je použitý PHP, ktorý podporuje väčšina unixových webových serverov.

3.1.5 Modulárnosť aplikácie

Aby mohla aplikácia byť rozšírená o ďalšie funkcie, musí obsahovať rozhranie cez ktoré je možné pridať tieto funkcie do aplikácie bez nutnosti zásahu do jej štruktúry. Z tohoto dôvodu je aplikácia rozdelená na viac nezávislých funkčných častí (programov) ktoré spolu komunikujú a tak vytvárajú celok (monitorovaciu aplikáciu).

3.1.6 Decentralizácia

V dnešnej dobe je veľmi dôležité vykonávať kontrolu dostupnosti z viacerých miest, pretože výpadok sa nemusí prejaviť v každom sieťovom segmente cez ktorý sa pripájajú klienti k poskytovanej službe. Preto je aplikácia navrhnutá ako klient-server riešenie, kedy na jednu službu môže byť nastavený monitoring z rôznych miest v sieti. Klient má možnosť pripojiť sa na každý z monitorovacích serverov a kontrolovať tak dostupnosť aplikácii z viacerých miest a odhaliť prípadný výpadok v niektorej z použitých dátových ciest.

3.2 Návrh

Táto podkapitola obsahuje návrh samotnej aplikácie pre monitorovanie, rozhrania pre pridávanie modulov a návrh niektorých modulov pre monitorovanie sieťových služieb. Aplikácia je rozdelná na viac samostatných funkčných častí. Základ tvorí klient-server aplikácia, ktorá sa stará o obsluhu nastavení a ovládanie monitoringu. Serverová časť vykonáva obsluhu samotného monitoringu, ukladanie dát. Kientská časť slúži na príjem a odosielanie nastavení, zastavovanie a spúšťanie modulov, príjem nazbieraných dát.

Serverová časť má na starosť správu nastavení, modulov, ich spúšťanie a zastavovanie, príjem požiadavkov od klientskej časti aplikácie a ich obsluhu.

3.2.1 Obslužná aplikácia - server (jadro,plánovač)

Hlavnou súčasťou monitorovacej aplikácie je obslužný program, bežiaci na pozadí systému, ktorý sa stará o správu nastavení, spúšťanie a zastavovanie monitorovacích modulov a obsluhu klientských požiadavkov pre správu monitoringu a požiadavky na dáta. Program obsahuje rozhranie pre pripojenie ďalších modulov pre monitorovanie, aby bolo možné ho rozšíriť podľa konkrétnych potrieb. Všetky nastavenia su ukladané v XML súboroch, aby bol program nezávislý na iných systemovych službách (napríklad mySQL) a tým pádom menej náchylný na výpadky.

3.2.2 Obslužná aplikácia – klient

Nemenej dôležitou súčasťou je aj klientská aplikácia, pomocou ktorej sa pristupuje k nazhromaždeným dátam z monitoringu, upravujú nastavenia samotného serveru a príslušiacich monitorovacích modulov. Klient je riešený ako jednoduchý konzolový program (ovladaný PHP skriptom webového rozhrania), ktorý preposiela požiadavky zadané v príkazovom riadku na monitorovací server a obsluhuje prijímané dáta.

3.2.3 Webové rozhranie

Webové rozhranie je jediná časť aplikácie, s ktorou prichádza do styku koncový uživateľ, preto je navrhnuté tak, aby sa dalo rýchlo pochopiť a efektívne používať. Je navrhnuté s ohľadom na modulárnosť systému a jednoduchosť ovládania. Prvým krokom pri používani je pripojenie k monitorovaciemu serveru pri ktorom sa načítajú konfigurácie modulov. Po úspešnom načítaní sú k dispozícii jednotlivé moduly a ich funkcie. Všetky funkcie čo sa týka nastavení, zobrazovaní výsledkov su implementované v každom module zvlášť, aby nenarastala zložitosť použitého

rozhrania medzi aplikáciou a modulom a aby nevznikali obmedzenia pre funkčnosť modulov (napríklad použoitím jedného presne definovaného typu prezentácie výsledkov).

3.2.4 Modul

Každý monitorovací modul obsahuje dve časti. Jednu ako aplikáciu pre server (monitor), ktorá beží nezávisle na jadre a druhú ako súčasť webového rozhrania, obsluhujúcu operácie s daným modulom. Forma nezávislej aplikácie bola zvolená z dôvodu vyššej odolnosti voči chybám ostatných modulov, prípadne samotného serveru. V prípade, že by každý modul bol použitý ako dynamicky pripájaná funkcia, mohlo by v prípade chyby v jednom module dôjst k zlyhaniu celej aplikácie (napríklad pri neoprávnenom prístupe do pamäti je automaticky jadrom operačného systému zastavený celý program). Toto riziko je minimalizované na jednu dymanicky pripájanú funkciu, z každého modulu, ktorá sa stará o prenos dát, ktoré modul generuje, ku klientovi. Server totiž spravuje len nastavenia, ktore majú z časti unifikovanú podobu, ale špecifické nastavenia modulov nespracúva – miesto toho pripája funkciu z dynamickej knižnice ktorá načíta nastavenia a na ich základe obslúži požiadavok na špeficifké dáta modulu. Každý modul implementuje metody spúšťania, zastavovania a určitý formát súboru s nastavením podľa rozhrania serveru, avšak samotný monitoring vykonáva vlastným spôsobom.

3.2.5 RRD

RRD databázu si spravuje každý modul zvlášť, stará sa o vytvorenie (príkaz `rrdtool create` [názov súboru] [parametre]), aktualizáciu (`rrdtool update [názov súboru] [hodnoty]`) a generovanie grafických výstupov (`rrdtool graph [názov súboru] [parametre grafu]`). RRD ako také ale nie je vynútený spôsob ukladania dát. Každý modul môže použiť hocijaký iný systém alebo prostriedok na ukladanie svojich výstupov (napr. mySQL databázu). Základná sada modulov po pridaní monitorovaného cieľa vytvorí dátový RRD súbor s predom definovanými parametrami (časový interval aktualizácie údajov v databáze, dĺžka obdobia uloženého v database, rozsahy jednotiek, atď.) do ktorého následne ukladá dáta. V prípade požiadavku na výstup od uživateľa sa tento dátový súbor prenesie ku klientovi, ktorý z neho generuje konkrétne výstupy podľa požiadavku uživateľa [1].

4 Implementácia

Táto kapitola sa zaoberá samotnou implementáciou monitorovacej aplikácie, použitými technikami monitorovania, problémami a ich implemenotvanými riešeniami.

4.1 Obslužná aplikácia – server

Server obsluhujúci monitorovacie moduly je implementovaný ako démon – aplikácia bežiaca na pozadí operačného systému.

4.1.1 Spustenie,inicializácia a ukončenie

Prvým krokom behu serveru je kontrola argumentov zadaných v príkazovom riadku. Server akceptuje jeden argument, ktorý môže nadobúdať hodnoty `start, stop, status`. V prípade, že je zadaný iný alebo žiadny argument, vypíše nápovedu k spusteniu a ukončí beh. Nápoveda má následujúci tvar: `usage: monsrv {start|stop|status}.`

Podľa zadaného argumentu program vykoná akciu – spustenie, zastavenie alebo zistenie aktuálneho stavu serveru. Po prijatí požiadavku na štart démona je volaná funkcia `action_start()`. Táto funkcia má za úlohu zistiť (pomocou funkcie `getppid()`), či server beží na pozadí (tj jeho parent id je 1) alebo je nutné beh programu presunúť do pozadia (parent id je rozdielne od 1). Ak už program beží na pozadí, je volaná funkcia `daemon_main()`, v opačnom prípade sa program replikuje pomocou funkcie `fork()` a pôvodná kópia (tzv. parent) ukončí svoj beh. Novo vzniknutá kópia (child) programu uzavrie všetky otvorené súborové deskriptory (`stdin, stdout, stderr`) a pokúsi sa o pridelenie exkluzívneho prístupu k súboru so zámkom. Ak sa zámok podarí prideliť, zapíše do súboru svoj PID.Ďalej maskuje signaly OS (tty, child, interrupt) a pokračuje funkciou `daemon_main()`. V prípade, že sa nepodarí získať exkluzívny prístup k súboru so zámkom, ukončí svoj beh (tj server uz beží, a nie je možné pokračovať).

Ak program dostal požiadavok na ukončenie, je volaná funkcia action_stop(), ktorá načíta aktuálne nastavenia serveru pomocou funkcie `get_config()`, zistí PID bežiaceho serveru, postupne vypne všetky bežiace moduly a nakoniec ukončí samotný server.

V prípade požiadavku na zistenie aktuálneho stavu je volaná funkcia `action_status()`. Funkcia sa pokúsi získať zapisovacie práva k súboru so zámkom. Pokiaľ má na súbor exkluzívny prístup bežiaci server (dôjde k zamietnutiu prístupu), vypíše na štandardný výstup správu o bežiacom serveri, načíta nastavenie modulov a prejde k postupnému zisťovaniu stavu každého bežiaceho modulu (resp. každého modulu ktorý by mal podľa nastavení byť spustený). V prípade, že sa

programu podarí získať prístup k súboru so zámkom, vypíše správu o nebežiacom serveri a ukončí svoju činnosť.

Ak je počas behu programu zachytený signál `SIGINT`, je spustená funkcia `signal_handler()` ktorá volá funkciu `action_stop()`.

4.1.2 Načítanie nastavení serveru

Načítanie nastavení serveru z XML súboru vykonáva funkcia `get_config()`. Prácu s XML zabezpečuje open source toolkit `tinyXML`. Funkcia načíta nastavenia z definovaného súboru pomocou volaní tinyXML a uloží ich do internej premennej uchovávajúcej potrebné parametre serveru (naslúchací port, logovanie udalostí, kľúč pre pripojenie klienta, počet modulov) a pre každý modul jeho názov a požadovaný stav (povolené alebo zakázané spustenie).

4.1.3 Vlastný beh serveru

Beh serveru po inicializácii obsluhuje funkcia `daemon_main()`. Prvým krokom je načítanie nastavení, vytvorenie a inicializácia socketu pre príjem spojení od klientskej časti aplikácie. Po vytvorení socketu a nastavení naslúchania nasleduje spustenie požadovaných modulov. Spustenie prebieha v cykle, kedy program postupne prechádza nastavenia. V prípade že modul na aktuálnom indexe má byť spustený, program zmení pracovný adresár podľa názvu modulu (`system('chdir modules/<názov modulu>');`), spustí modul príkazom `./module start` a vykoná návrat do pôvodného pracovného adresára (`system('chdir ../../');`).

Po spustení požadovaných modulov prejde server do štádia príjmu spojení na vytvorenom sockete. Po vytvorení spojenia pomocou funkcie `accept`() je uložené čislo socketu a predané ako parameter vláknu `clientThread`, ktoré zabezpečuje načítanie a následnú obsluhu požiadavku od klienta.

4.1.4 Obsluha požiadavku klienta

Funkcia clientThread spustená ako samostatné vlákno (tj neblokujúca ďaľšie pripojenie) obsluhuje príjem požiadavku od klienta a odoslanie odpovedi klientovi. Prvým krokom je získanie exkluzívneho prístupu k zámku (mutexu), ktorý zaručí, že nevznikne kolízia medzi požiadavkami klienta (ako napríklad zápis a čítanie nastavení naraz). Nálseduje nadviazanie spojenia s klientom pomocou príkazu `KEY` nasledovaného prístupovým kľúčom. Po kontrole kľúča nasleduje načítanie príkazu zo socketu a vykonanie jeho obsluhy. Popis jednotlivých príkazov sa nachádza v ďaľšej podkapitole, pre každý príkaz má server jednu obslužnú funkciu ktora vykoná požadovanú akciu. Výnimkou je príkaz `GETDATA`, u ktorého je volaná funkcia z dynamicky linkovanej knižnice, ktorá sa stará o odoslanie dát modulu klientovi.

4.1.5 Komunikačný protokol

Komunikačný protokol pre účely výmeny dát medzi klientom a monitorovacím serverom je jednoduchý dvoj až trojkrokový protokol. Prvým krokom po naviazaní spojenia je odoslanie príkazu KEY a komunikačného kľúča, za ktorým následuje jeden z príkazov pre správu serveru.

Klient	Server	Funkcia
Popis		
KEY□[kluc] □	-	-
Zahájenie spojenia po ktorom následuje ďalší príkaz. V prípade nesprávneho kľúča server ukončí spojenie.		
LISTALL□	[pocet suborov]□[nazov suboru]□ [dlzka]□[data]…	ListAllConf()
Načítanie nastavení všetkých modulov a ich odoslanie klientovi.		
LISTG□	[dlzka] □[data]	ListGlobalConf()
Načítanie nastavení serveru a odoslanie klientovi.		
LISTM□[nazov modulu] □	[dlzka] □[data]	ListModuleConf()
Načítanie a odoslanie konfiguračného súboru zadaného modulu.		
SETMOD□[nazov modulu]□[dlzka]□[data]	-	SetModuleConf()
Príjem a zapísanie konfiguračného súboru zadaného modulu.		
SETSRV□[dlzka] □[data]	-	SetServerConf()
Príjem a zapísanie konfigurácie serveru.		
GETDATA□[nazov modulu] □	[pocet suborov] □[nazov suboru]□ [data]…	GetModuleData()
Načítanie a odoslanie dátovych súborov príslušiacich k zadanému modulu, obsluhu tejto funkcie vykonáva dynamicky pripájaná funkcia, ktorá odošle na socket počet súborov a ich jednotlivé dáta v stanovenom formáte.		
RESTARTM□[nazov modulu] □	-	ModuleRestart()
Načíta názov modulu a ten reštartuje.		
RESTARTA□	-	ModuleRestartAll()
Reštartuje všetky moduly podľa nastavení serveru.		
Poznámka: znak □ je oddeľovač príkazov, ASCII znak s číslom 0x1 (‘\x01’)		

Tabuľka 1 Prehľad príkazov komunikačného protokolu.

4.2 Obslužná aplikácia – klient

Klientský program je rozhraním k monitorovacej aplikácii, cez ktoré sa vykonávaju všetky akcie spojené so správou monitoringu. Program pre príkazový riadok je navrhnutý tak aby mohol byť spúšťaný z PHP skriptu, tj. je neinteraktívny. Všetky parametre pre svoj beh získava z argumentov príkazového riadka. Argumenty, ktoré program akceptuje sú uvedené v tabuľke 2.

-h [hostname]	Názov hostu alebo adresa, na ktorej naslúcha server.
-p [port]	Port na ktorom naslúcha server, nepovinný parameter.
-k [key]	Kľúč pre pripojenie k serveru
-c [command]	Príkaz, ktorý sa má vykonať

`-m [module]`	Názov modulu, na ktorom sa má vykonať príkaz (`main` pre globálne nastavenia serveru, `all` pre všetky moduly)
`--help`	Zobrazenie nápovedy

Tabuľka 2 Zoznam argumentov podporovaných klientskou aplikáciou.

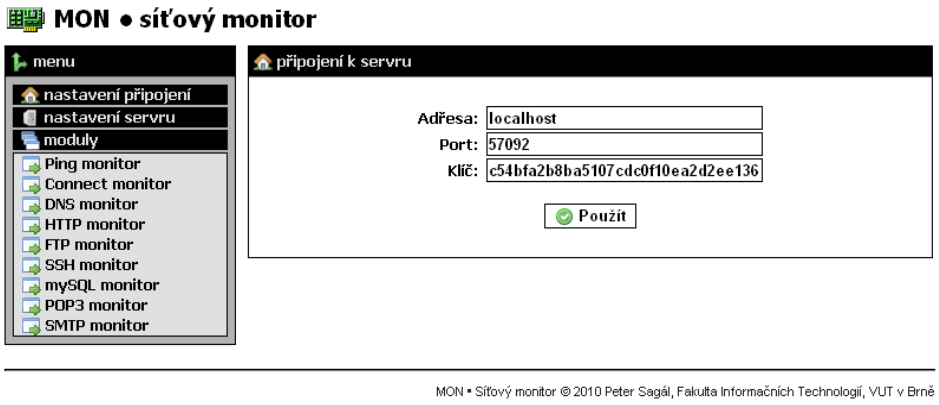
V prípade, že nie je uvedený niektorý z povinných parametrov, program vypíše chybové hlásenie a skončí. Zoznam možných príkazov je v tabuľke 3.

`getconf`	Príjem konfiguračného súboru (súborov)
`setconf`	Odoslanie konfiguračného súboru (súborov)
`getdata`	Príjem dátových súborov
`restartmodule`	Reštart modulu (modulov)

Tabuľka 3 Prehľad podporovaných príkazov klientskej aplikácie

Program po načítaní všetkých povinných argumentov vytvorí spojenie ku serveru, odošle príkaz pre naviazanie komunikácie a vykoná akciu danú argumentom príkazového riadka. Konfiguračné alebo dátové súbory ukladá do definovaných adresárov podľa názvu modulu (`[koreňový adresár stránok/modules/<názov modulu>]`). Pre správny beh programu je nutné mať správnu a kompletnú adresárovú štruktúru zhodnú so štruktúrou serveru, tzn. webové rozhranie musí obsahovať všetky podzložky v adresári `modules` zhodné so serverom.

4.3 Webové rozhranie



Obrázok 1 Webnové rozhranie aplikácie

Webové rozhranie (Obr 1.) je realizované pomocou skriptovacieho jazyka PHP s použitím XML toolkitu simplexml. Na jeho správny beh je nutné aby koreňový adresár webstránok obsahoval klienta pre pripojenie k monitorvaciemu serveru a bol nainštalovaný rrdtool, pomocou ktorého sa z dát modulov vytvárajú grafické výstupy pre zobrazenie na stránkach. Pri implementácii webového

rozhrania bola pre spríjemnenie vzhľadu použitá voľne dostupná sada ikon z webstánky http://famfamfam.com. Kontrola prístupu do webového rozhrania je riešená cez nastavenia súboru `.htaccess` na HTTP autentifikáciu skrz súbor `.htpasswd`, v ktorom je uložené užívateľské meno a hash hesla. Tento spôsob bol použitý ako náhrada cookies, ktoré môžu spôsobovať problémy s kompatibilitou a stabilitou behu aplikácie. Na prístup k webovému rozhraniu je z tohoto dôvodu nutné použit prehliadač, ktorý podporuje HTTP autentifikáciu.

Po prihlásení užívateľa sa zobrazí stránka s nastavením pripojenia (Obr. 2) k monitorovaciemu serveru, automaticky sa na nej načítajú posledné použité nastavenia, tj. nieje nutné pri každom príustupe nanovo zadávať údaje pre pripojenie k serveru.

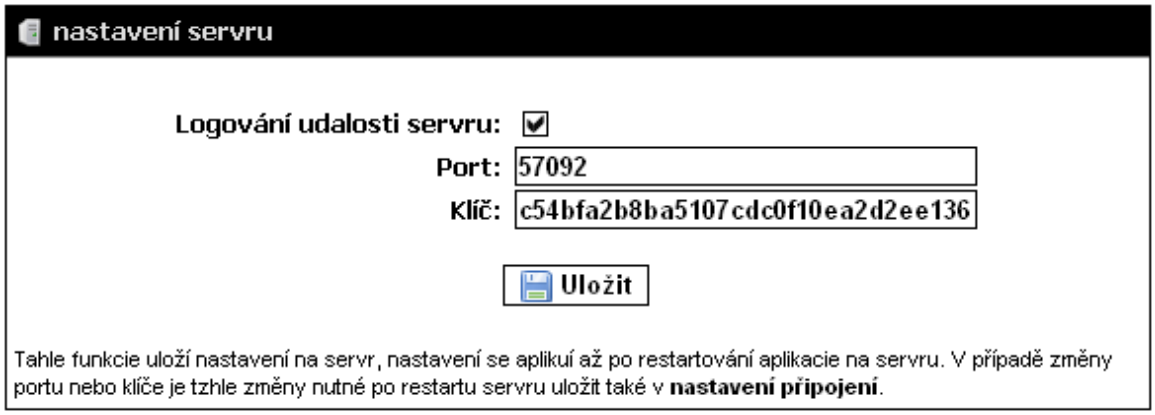
V ľavej časti stránky sa nachádza menu, ktoré zobrazuje položky s nastavením pripojenia k serveru, samotné nastavenia serveru a odkazy na stránky jednotlivých zapnutých modulov.

4.3.1 Nastavenie pripojenia

Obrázok 2 Nastavenie pripojenia k serveru

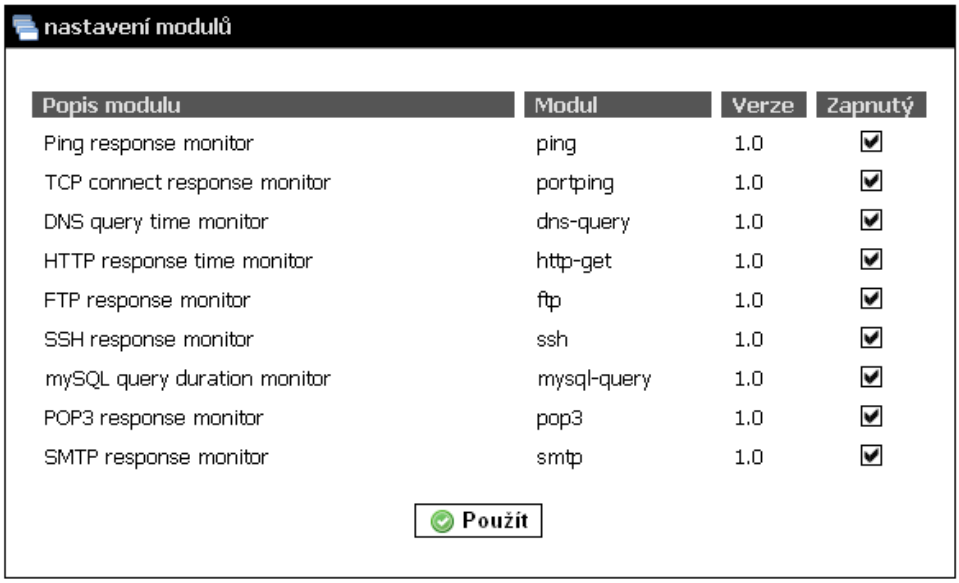
Stránka s nastavením pripojenia (Obr. 2) k serveru ponúka možnosť nastaviť tri parametre pripojenia – názov serveru pre pripojenie, port na ktorom server prijíma spojenia a sieťový kľúč pre pripojenie. Tieto nastavenia sa ukladajú do lokálneho súboru client.xml, pred uložením sa vykoná kontrola existencie zadaného názvu serveru pomocou funkcie `gethostbyname()`, rozsahu zadaného portu (číslo portu musí patriť do intervalu 1-65536) a obsahu kľúča (kľúč nesmie byť prázdny). V prípade chybového stavu (napríklad server neexistuje) je zobrazená chybová správa a nastavenia nebudú uložené.

4.3.2 Nastavenia serveru



Obrázok 3 Nastavenia serveru

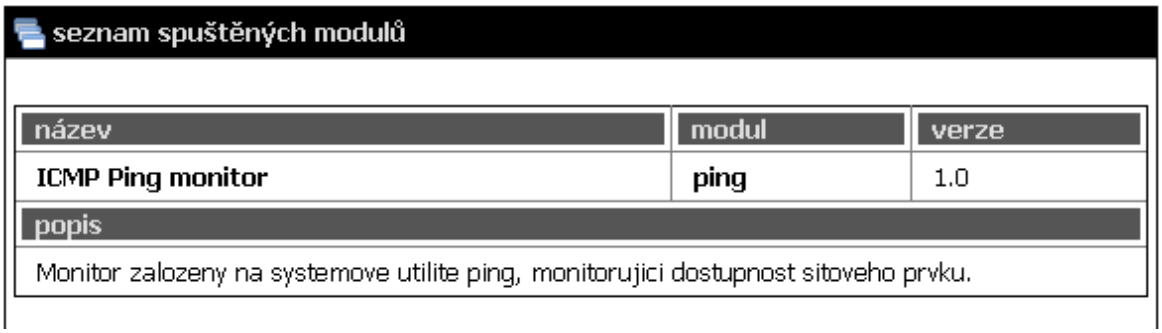
Stránka s nastavením serveru (Obr. 3) ponúka možnosti úpravy konfigurácie serveru – zapnutie logovania udalostí na serveri (vhodné pri odhaľovaní chýb serveru), nastavenie portu na ktorom server prijíma spojenia a sieťový kľúč použitý pre pripojenie klienta. Tieto nastavenia sú po odoslaní formulára kontrolované (rozsah portu, obsah kľúča) a v prípade chyby je zobrazené hlásenie, pokiaľ sa žiadna chyba nenašla, nastavenia sú uložené do súboru config.xml a odoslané na server príkazom: `./moncl –h [hostname] –p [port] –k [kľúč] –c setconf –m main`. Tieto nastavenia sa použijú pri novom spustení serverovej aplikácie. Samotný reštart aplikácie je nutné vykonať lokálne na serveri príkazom: `./mon stop && ./mon start` a podľa nových nastavení patrične upravit vlastnosti pripojenia k serveru.



Obrázok 4 Nastavenie spúšťania modulov

Stránka ďalej obsahuje zoznam nainštalovaných modulov (Obr. 4) a možnosť ich zapnúť / vypnúť. Po odoslaní formuláru sa upraví lokálna kópia súboru s nastavením serveru a tá sa potom odošle na server pomocou príkazu: `./moncl -h [hostname] –p [port] –k [kľúč] –c setconf –m all`. Po odoslaní nastavení sa pomocou klienta pošle serveru príkaz na reštart všetkých modulov podľa nových nastavení: `./moncl –h [hostname] –p [port] –k [kľúč] –c restartmodule –m all`.

4.3.3 Moduly



Obrázok 5 Stránka zobrazujúca spustené moduly s bližším popisom

Stránka obsahuje zoznam spustených modulov a ich bližší popis, verziu a presný názov (Obr. 5). Podstránky spustených modulov sú linkované v menu v skupine moduly. Samotné podstránky budú priblížené v následujúcej podkapitole.

4.4 Monitorovacie moduly

Monitorovacia aplikácia je z dôvodu možnosti ďalších rozšírení navrhnutá modulárne. Jedným z dôvodov je odolnosť voči chybám modulov, v prípade výpadku jedného modulu ostatné pracujú bez prerušenia. Moduly pracujú aj v prípade poruchy obslužného programu, keďže sú spúšťané ako samostatné aplikácie. Výpadok obslužného programu spôsobí nedostupnosť služieb poskytovaných webovým rozhraním, avšak monitoring bude prebiehať ďalej. Povinnou súčasťou každého modulu je konfiguračný XML súbor obsahujúci povinné kontajnery s dátami – krátky názov modulu, dlhý názov modulu, titulok pre menu webového rozhrania a popis modulu. Ostatné nastavenia sú závislé na konkrétnom module. Ďalšou povinnou súčasťou je spustiteľný súbor s modulom a názvom `module`, knižnica `library.so` obsahujúca dznamicky pripájanú funkciu `send_data()`. Povinná súčasť v rámci webového rozhrania je stránka `main.php` v zložke `[názov modulu]/pages` ktorá je priamo odkazovaná z menu rozhrania.

Vlastné monitorovanie vykonáva každý modul samostatne na základe samostatných špecifických nastavení. Týmto nie je modul obmedzený na vykonaváníe aktívneho alebo pasívneho monitoringu.

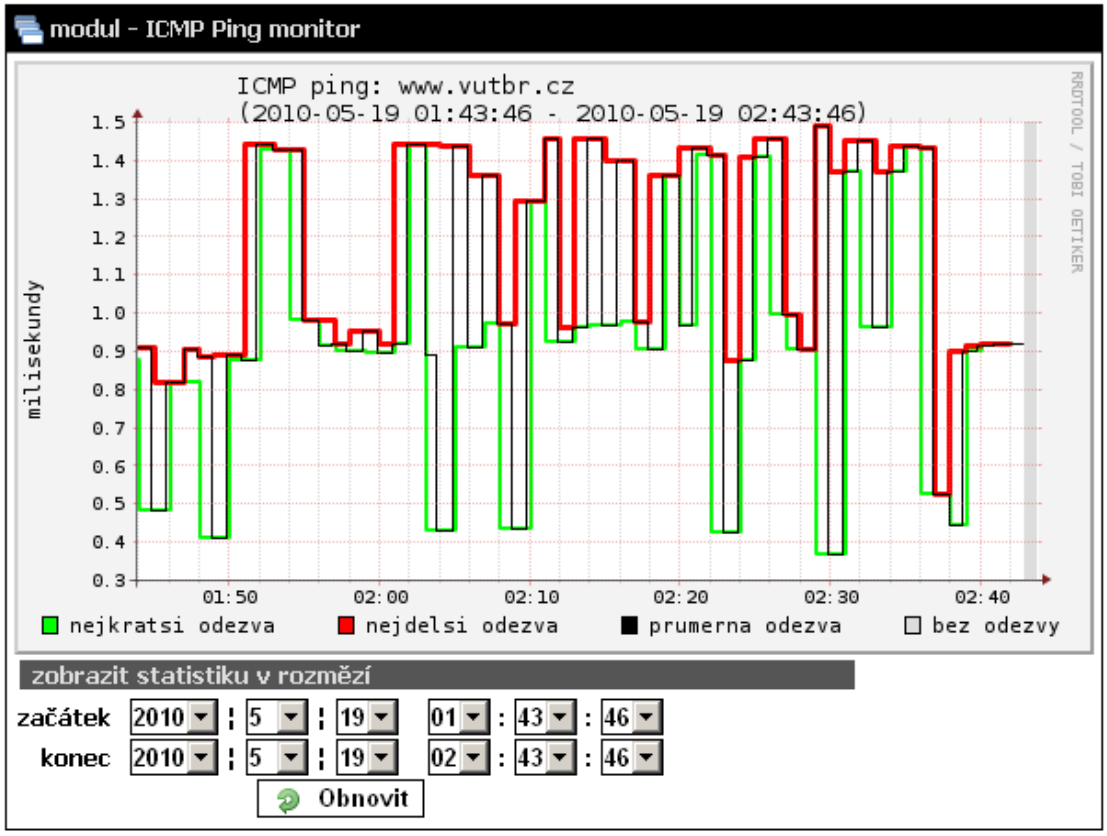
Každý modul musí implementovať jeden povinný argument príkazového riadka pomocou ktorého sa ovláda jeho činnost. Akceptované hodnoty argumentu sú uvedené v tabuľke 4.

`start`	Spustenie modulu na pozadí
`stop`	Zastavenie modulu
`status`	Vypísanie aktuálneho stavu modulu

Tabuľka 4 Povinne podporované hodnoty argumentu príkazového riadka modulu.

Modul po prijatí argumetu vykoná akciu pomocou algoritmu popísaného v kapitole 4.1.1 . V ďaľších podkapitolách sú uvedené konkrétne postupy pre monitorovanie použité konkrétnymi modulmi a ich výstup vo webovom rozhraní.

Všetky implementované moduly používajú rovnakú schému webového rozhrania. Hlavná stránka webového rozhrania obsahuje zoznam monitorovaných adries so štvoricou akcií pre každú položku zoznamu. Prvou je zobrazenie grafov za prednastavené časové obdobia (hodina, deň, týždeň, mesiac), druhou je možnosť zobraziť graf podľa zadaných časových parametrov, treťou akciou je úprava parametrov monitoringu a poslednou je odstránanie monitoringu. Hlavná stránka ďalej obsahuje formulár pre pridanie monitoringu podľa zadaných parametrov. Pri ukladaní nastavení monitoringu prebieha kontrola vstupných údajov v závislosti na konkrétnom zadávanom parametri. Po bezchybnom priebehu kontroly sú nové dáta pridané / uložene v lokálnej kópii nastavení modulu a súbor s nastaveniami je následne odoslaný serveru. Po nahraní nastavení na server je modul reštartovaný, aby sa tieto nové nastavenia ihneď použili.



Obrázok 6 Zobrazenie grafického výstupu modulu ping

Stránka zobrazujúca grafy (Obr. 6) (stat.php, stat_custom.php) načíta dáta zo serveru pre daný modul – volaním aplikácie klienta `./moncl -h hostname -p port –k kluc –c getdata –m nazov_modulu`. Následne vygeneruje grafy podľa zadaných hodnôt časových údajov (buď predvolených alebo zadaných uživateľom) a tie následne zobrazí. Samotné generovanie grafov obstaráva utilita rrdgraph (volanie: `rrdtool graph [názov súboru.png] [parametre pre generovanie grafu]`), ktorá je súčasťou toolkitu RRDTool. Moduly, ktoré vykonávajú viac monitorovacích cyklov v jednom kroku generujú grafy obsahujúce minimálne, maximálne a priemerné odozvy, ostatné moduly generujú len jeden graf s hodnotami odoziev.

4.4.1 ICMP Ping

Modul vykonáva základný test dostupnosti pomocou systémovej utility ping. Po načítaní nastavení zo súboru module.xml vytvorí vlákna, ktoré obsluhujú jednotilvé monitorované hosty. Podľa zadaného časového úseku opakovania vlákno spúšťa monitorovací skript check.sh, ktorý spustí príkaz ping s požadovanými parametrami a získane výsledky uložíi do RRD databázy. Zistenie výsledkov je realizované pomocou textových utilít grep a awk. Skriptu program predáva názov monitorovaného serveru, maximálnu dĺžku čakania na odozvu a počet pokusov, ktoré sa majú vykonať.

4.4.2 TCP Spojenie

Modul testuje dostupnosť sieťového TCP portu na zadanej adrese pomocou funkcie connect(). Meria čas, za ktorý funkcia prebehne (tj. Čas, za ktorý sa podarí vytvoriť spojenie na zadanú adresu a port) a ten následne pomocou utility rrdupdate ukladá do RRD databázy.

Po spustení modulu a prenesení do pozadia OS program načíta nastavenia zo svojho konfiguračného súboru – hostname a port pre pripojenie. Potom pre každú monitorovanú adresu vytvorí vlákno, ktoré obsluhuje vlastný monitoring – v daných časových intervaloch spúšťa funkciu check_port(), ktorá vykoná pripojenie a meria čas, za ktorý sa spojenie vytvorí. Namerané údaje potom pomocou volania programu rrdtool uloží do databázy.

4.4.3 HTTP

Modul pre kontrolu dostupnosti webových stránok je implementovaný na podobnom princípe ako modul pre ICMP ping. Vlastná aplikácia po spustení a prechode do pozadia načíta nastavenia z XML súboru, pre každú monitorovanú URL vytvorí vlákno, ktoré sa ďalej stará o monitorovanie. Vlákno v pravidelných intervaloch spúšťa skript, ktorý vykoná pripojenie a stiahnutie danej stránky pomocou utility wget. Pred spustením programu wget uloží časovú značku začiatku pripájania, po ukončení programu uloží druhú časovú značku a nameraný rozdiel potom pomocou programu rrdupdate uloží do rrd databázy.

4.4.4 FTP

Modul na monitorovanie služby ftp je ďalším modulom, ktorý používa pri monitoringu externé aplikácie a skriptovacie jazyky. Modul ako predošlé popísané moduly po spustení načíta nastavenia, pre každý monitorovaný host spustí vlákno, ktoré ďalej obsluhuje monitorovanie. V prípade služby ftp je nutné interaktívne prihlásenie a zadávanie príkazov. Monitorovací skript check.sh preto používa k prihláseniu ďalší skript try.sh, ktorý je spúšťaný v interprete expect. Interpret expect je navrhnutý na skriptovanie interaktívnych činností. Tento skript sa pripojí na server pomocou príkazu ftp, zadá prihlasovacie údaje a odloguje sa zo serveru. Skript check.sh stopuje čas, za ktorý skript try.sh prebehne a namerané hodnoty odoziev ukladá do RRD databázy.

4.4.5 SSH

Ako už bolo spomenuté v popise modulu ftp, aj služba ssh vyžaduje interaktívne prihlásenie, preto sa situácia opakuje ako v predošlom prípade. Modul po spustení načíta nastavenia monitoringu, spustí vlákna, ktoré v zadaných časových intervaloch spúšťajú skript check.sh. Skript uloží pred spustením try.sh časovú značku, vykoná skript try.sh a po jeho ukončení uloží druhú časovú značku. Rozdiel

týchto časov je dĺžka odozvy, ktorú uloži do RRD databázy. Z dôvodu bezpečnostného rizika odporúčam pomocou tohoto skriptu monitorovať len účty vytvorené k tomuto účelu.

4.4.6 mySQL

Modul pre monitorovanie mySQL databázy na rozdiel od ostatných modulov používa pre monitorovanie databázy PHP skript, ktorý je spúšťaný rovnakým spôsobom ako shellovské skripty v ostatných moduloch – tj. po spustení a načítaní nastavení sa vytvoria vlákna, ktoré v zadaných časových intervaloch spúšťajú monitorovací PHP skript. PHP skript sa pomocou funkcie mysql_connect90 pripojí k zadanej databaze, vyberie databazu podľa nastavení a vykoná zadaný SQL dotaz. Čas, za ktorý prebehne pripojenie, vykonanie dotazu a odpojenie od serveru je ukladaný do RRD databázy.

4.4.7 DNS

Modul monitorujúci dĺžku odozvy DNS servera používa na zistenie dĺžky odozvy utilitu dig. Modul spustí vlákna podľa nastavení monitoringu. Vlákna v zadaných časových interavloch spúšťajú skript check.sh, ktorý spúšťa pre každý zo zadaných hostnamov prikaz dig, ktorý získava odozvy zadaného DNS serveru. Program dig, podobne ako ping, obsahuje vo svojom výpise dĺžku času, za ktorý dostal odpoveď. Skript túto hodnotu z výpisu extrahuje pomocou textových utilít grep a awk zreťazených do tzv pipe. Získané hodnoty su zapisované do RRD databázy.

4.4.8 POP3

Modul monitorujúci odozvy poštového POP3 servera funguje bez externých skriptov. Pri analýze možností monitorovania som nenarazil na vhodnú utilitu pre monitorovanie. Modul teda implementuje jednoduchého POP3 klienta, ktorý je spúšťaný v pravidelných intervaloch pre každý monitorovaný poštový server. Klient vykoná štandardné pripojenie podľa špecifikácie protokolu [11], prihlásenie klienta, odošle príkaz HELP a odloguje sa zo serveru. Dĺžky časových úsekov nutných na vykonanie týchto operácií ukladá do RRD databazy.

4.4.9 SMTP

Monitoring smtp serveru vykonáva podobne ako pop3 monitoring vlastný smtp klient, ktorý vytvorí spojenie so serverom na TCP porte 25, odošle prednastavenú identifikáciu (príkaz ‚HELO test'), pošle príkaz HELP a odloguje sa zo serveru prikazom QUIT [12]. Čas nutný na vykonanie tejto postupnosti príkazov ukladá do RRD databázy. V prípade chyby počas ľubovoľného kroku ukončuje klient spojenie a do databázy sa pre daný časový okamih uloží hodnota ‘unknown‘ signalizujúca výpadok služby.

5 Záver

Cieľom tejto bakalárskej práce bolo preštudovať možnosti monitorovania odoziev siete, sieťových prvkov a aplikácií so zameraním na porovnanie rozdielov medzi aktívnym a pasívnym monitorovaním, na základe analýzy požiadavkov na monitorovanie navrhnúť systém, ktorý bude schopný monitorovať odozvy siete a sieťových služieb.

Na monitorovanie siete sa používajú dve metodiky. Pasívne monitorovanie prevádzky a aktívne posielanie požiadavkov na sieť a aplikácie. Pasívne monitorovanie je vhodné na zisťovanie výkonu sieťových aplikácií, ale keďže sa pri ňom analyzujú reálne dáta prúdiace sieťou, nie je možné odhaliť výpadok pred tým ako tento výpadok zasiahne klienta. Z tohoto dôvodu som sa rozhodol pre implementáciu druhého spôsobu monitorovania a to aktívneho monitoringu, ktorý poskytuje rýchlejšiu odozvu na výpadok služby a tým umožňuje rýchlejšie lokalizovanie a odstránenie problému. Aktívne monitorovanie používa na zisťovanie stavu simuláciu uživateľskych požiadavkov, ktoré odosiela aplikáciam a na základe analýzy odpovedí vyhodnocuje situáciu.

Na simuláciu uživateľských požiadavkov som pri implementácii použil utility bežne sa vyskytujúce v unixovych systémoch – ping, wget, ftp, ssh, dig. V niektorých prípadoch, keď systém neponúkal utilitu vhodnú na monitorovanie služby som vytvoril vlastného klienta pre účely monitorovania, ktorý obsahuje len obmedzenú funkčnost nutnú pre potreby monitoringu. Namerané hodnoty časových odoziev su ukladané a spracovávané programom RRDTool, ktorý poskytuje nástroje na ukladanie štatistických dát a ich následnu prezentáciu vo forme grafického zobrazenia.

Ovládanie celého navrhnutého systému zabezpečuje webové rozhranie, ktoré bolo implementované s ohľadom na budúcu rozšíritieľnost a jednoduchosť ovládania. Prihlásený uživateľ má plný prístup ku všetkým prostriedkom poskytovaným aplikáciou, od nastavení serveru, webového rozhrania až po nastavenia jednotlivých modulov vykonávajúcich vlastné monitorovanie.

Keďže systém bol navrhnutý s ohľadom na ďaľšie rozširovanie, pridávanie funkcionality je jednoduchou úlohou, pri ktorej netreba nijak zasahovať do systému. Aplikácia poskytuje rozhranie pre beh a ovládanie samostatných monitorovacích modulov. Tým pádom môže byť rozšírená napríklad o pasívny monitoring, ktorý je vhodným doplnkom aktívneho monitorovania.

Vzhľadom na to, že aplikácia je len návrhom systému, niektoré možnosti nastavenia boli pred uživateľom skryté. Jedná sa hlavne o nastavenie dĺžky monitorovacích cyklov, ktoré by v prípade, že by ich uživateľ mohol meniť, museli prechádzať špeciálnou kontrolou na zabránenie zahltenia systému – napríklad keby timeout ping paketu presiahol dĺžku cyklu (boli by odosielané nové pakety ešte pred prijatím predošlých), mohlo by nastať zahltenie aplikácie prípadne sieťového ovládača.

Ďaľším návrhom na zapracovanie je možnosť ovládať viac monitorovacích serverov súčasne – napríklad pri decentralizácii monitoringu by bolo možné porovnávať výsledky z rôznych monitorovacích zdrojov. Keďže aplikácia je modulárna, bolo by vhodné mať možnosť nahrávať moduly priamo cez webové rozhranie a zapájať ich do monitoringu bez nutnosti reštartovať celú monitorovaciu aplikáciu.

Aj napriek spomenutym nedostatkom systém počas testovania pracoval spoľahlivo, generovane výstupy zodpovedali reálnej situácii na sieti. Pravidelné výkyvy v dĺžke odozvy spôsobovalo pravidelné zaťažovanie serveru (generovanie systémovej štatistiky, taktiež pomocou RRDtool), na ktorom bol systém testovaný.

Literatura

[1] *RRDtool Documentation.* 2009. [online] Dostupné z URL: http://oss.oetiker.ch/rrdtool/doc/index.en.html (máj 2010).

[2] *PHP Manual*. [online] Dostupné z URL: http://www.php.net/manual/en/ (máj 2010).

[3] *TinyXML Documentation*. [online] Dostupné z URL: http://www.grinninglizard.com/tinyxmldocs/index.html (máj 2010).

[4] *Unix Daemon Server Programming*. [online] Dostupné z URL: http://www.enderunix.org/documents/eng/daemon.php (máj 2010).

[5] Case, J., Fedor, M., Schoffstall, M., Davin, J.: *A Simple Network Management Protocol.* RFC 1157, 1990. [online] Dostupné z URL: http://www.ietf.org/rfc/rfc1157.txt (máj 2010).

[6] Postel, J.: *Internet Control Message Protocol.* RFC 792, 1981. [online] Dostupné z URL: http://www.ietf.org/rfc/rfc0792.txt (máj 2010).

[7] Clymer, Ch.: *Expect: Automating Your Shell*. [online] Dostupné z URL: http://chrisclymer.com/articles/expect/ (máj 2010)

[8] *GNU Wget 1.12 Manual*. [online] Dostupné z URL: http://www.gnu.org/software/wget/manual/wget.html (máj 2010)

[9] *dig manual page*. [online] Dostupné z URL: http://members.shaw.ca/nicholas.fong/dig/dig-man.html (máj 2010)

[10] Cooper, M.: Advanced BASH-Scripting Guide. 2010. [online] Dostupné z URL: http://tldp.org/LDP/abs/html/ (máj 2010)

[11] Myers, J., Rose, M.: *Post Office Protocol – Version 3*. RFC 1939, 1996. [online] Dostupné z URL: http://www.ietf.org/rfc/rfc1939.txt (máj 2010)

[12] Postel, J.: *Simple Mail Transfer Protocol*, RFC 821, 1982. [online] Dostupné z URL: http://www.ietf.org/rfc/rfc0821.txt (máj 2010)

Seznam příloh

Příloha 1. CD so zdrojovými kódmi aplikácie a elektronickou verziou bakalárskej práce